# Text Processing

- Comma Sepearated values (CSV)
- JavaScript Object Notation(JSON),
- Python and XML

# CSV

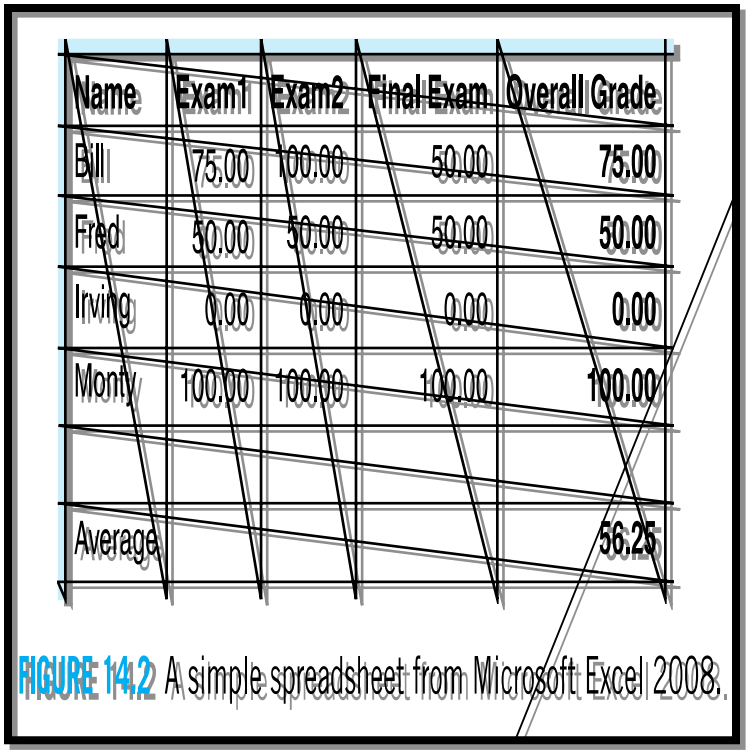
- The csv module is useful for working with **data exported from spreadsheets and databases into text files formatted with fields and records**, commonly referred to as *comma-separated value* (CSV) format because **commas are often used to separate the fields in a record.**

- CSV is an short form for Comma Separated Values.
- The CSV format is a very common data format used by various programs and applications as a format to export or import data.
- Most spreadsheet and database applications offer export and import facilities for CSV files.
- **CSV files generally store tables of data that contain fields stored within records.**
- **Each field in the record is separated by a character within the CSV file** and **generally new records start on a new line.**
- This is what a CSV file looks like that has been exported from Excel:

- CSV files are simple, lacking many of the features of an Excel spreadsheet. For example, CSV files:
- **Don't have types for their values—everything is a string**
- **Don't have settings for font size or color**
- **Don't have multiple worksheets**
- **Can't specify cell widths and heights**
- **Can't have merged cells**
- **Can't have images or charts embedded in them**

- The advantage of CSV files is simplicity.
- CSV files are widely supported by many types of programs, can be viewed in text editors (including IDLE's file editor), and are a straightforward way to represent spreadsheet data.
- The CSV format is exactly as advertised: **It's just a text file of comma-separated values.**

# Spread sheet and corresponding CSV file



| Name | Exam1 | Exam2 | Final Exam | Overall Grade |
|---|---|---|---|---|
| Bill | 75.00 | 100.00 | 50.00 | 75.00 |
| Fred | 50.00 | 50.00 | 50.00 | 50.00 |
| Irving | 0.00 | 0.00 | 0.00 | 0.00 |
| Monty | 100.00 | 100.00 | 100.00 | 100.00 |
| | | | | |
| Average | | | | 56.25 |

FIGURE 14.2 A simple spreadsheet from Microsoft Excel 2008.

```
Name,Exam1,Exam2,Final Exam,Overall Grade
Bill,75.00,100.00,50.00,75.00
Fred,50.00,50.00,50.00,50.00
Irving,0.00,0.00,0.00,0.00
Monty,100.00,100.00,100.00,100.00

Average,,,,56.25
```

# CSV format

- As simple as that sounds, even CSV format is not completely universal ,different apps have small variations
- Python provides a module to deal with these variations called the csv module
- **This module allows you to read spreadsheet info into your program**
- We load the module in the usual way using import:

```
>>> import csv
```

# writer

- import csv **#import modules**
- f=open("sanmple.csv","w+")
- a=["Name","Address","phone"],["ravi","amreli","12345"], ["raviraj","rajkot","132345"]
- w1=csv.writer(f) **#create writer object**
- w1.writerow(a) **#write data**
- print("data written")
- f.close() **#close file**

# reader

```
import csv
#reading CSV data
f=open("sanmple.csv","r+")
r1=csv.reader(f)
for row in r1:
    print(row)
f.close()
```

# Quoting

- There are four different quoting options, defined as constants in the csv module.
- QUOTE_ALL Quote everything, regardless of type.
- QUOTE_MINIMAL Quote fields with special characters (anything that would confuse a parser configured with the same dialect and options). This is the default
- QUOTE_NONNUMERIC Quote all fields that are not integers or floats. When used with the reader, input fields that are not quoted are converted to floats.
- QUOTE_NONE Do not quote anything on output. When used with the reader, quote characters are included in the field values

# JavaScript Object Notation (JSON)

- JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange.
- Conventions used by JSON are known to programmers, which include C, C++, Java, Python, Perl, etc.
- JSON stands for JavaScript Object Notation.
- The format was specified by **Douglas Crockford.**
- It was designed for human-readable data interchange.
- It has been extended from the JavaScript scripting language.

# Characteristics of JSON

- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.

# JSON is Not

- Overly Complex
- A “document” format
- A markup language
- A programming language

# JSON like XML

- Plain text formats
- “Self-describing“ (human readable)
- Hierarchical (Values can contain lists of objects or values)
- Lighter and faster than XML
- JSON uses typed objects. All XML values are type-less strings and must be parsed at runtime.
- Less syntax, no semantics
- Properties are immediately accessible to JavaScript code

# JSON Object Syntax

- Unordered sets of name/value pairs
- Begins with **{** (left brace)
- Ends with **}** (right brace)
- Each name is followed by **:** (colon)
- Name/value pairs are separated by **,** (comma)

# JSON example

```
employeeData = {
   "employee_id": 1234567,
   "name": "Jeff Fox",
   "hire_date": "1/1/2013",
   "location": "Norwalk, CT",
   "consultant": false
};
```

# Arrays in JSON

- An ordered collection of values
- Begins with **[** (left bracket)
- Ends with **]** (right bracket)
- Name/value pairs are separated by **,** (comma)

```
employeeData = {
   "employee_id": 1236937,
   "name": "Jeff Fox",
   "hire_date": "1/1/2013",
   "location": "Norwalk, CT",
   "consultant": false,
  "random_nums": [ 24,65,12,94 ]
};
```

# Data Types: Object & Array

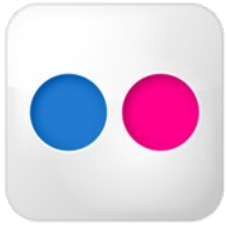
- Objects: Unordered key/value pairs wrapped in { }
- Arrays: Ordered key/value pairs wrapped in [ ]

# How & When to use JSON

- Transfer data to and from a server
- Perform asynchronous data calls without requiring a page refresh
- Working with data stores
- Compile and save form or user data for local storage

# Where is JSON used today?

- Anywhere and everywhere!




And many, many more!

# JSON

- In Python, the json module provides an API similar to convert in-memory Python objects to a serialized representation known as JavaScript Object Notation (JSON) and vice-a-versa.

- **Encode Python objects as JSON strings, and decode JSON strings into Python objects**

# Encode Python objects as JSON strings

```
json.dump(obj, fp,
          skipkeys=False,
          ensure_ascii=True,
          check_circular=True,
          allow_nan=True,
          cls=None,
          indent=None,
          separators=None,
          default=None,
          sort_keys=False, **kw)
```

- if you want to dump the JSON into a file/socket or whatever, then you should go for **dump().**
- If you **only need it as a string** (for printing, parsing or whatever) **then use dumps() (dump string)**

| Python | JSON |
| --- | --- |
| dict | object |
| list, tuple | array |
| str | string |
| int, float, int- & float-derived Enums | number |
| True | true |
| False | false |
| None | null |

- **Options :**
- The default value of **skipkeys** is False. If skipkeys is True, then dict keys that are not of a basic type (str, int, float, bool, None) will be skipped instead of raising a TypeError.
- The json module always produces str objects, not bytes objects. Therefore, fp.write() must support str input.
- **If ensure_ascii** is True (the default), the output is guaranteed to have all incoming non-ASCII characters escaped. If ensure_ascii is False, these characters will be output as-is.
- The default value of **check_circular** is True. If check_circular is False, then the circular reference check for container types will be skipped and a circular reference will result in an OverflowError.
- The default value of **allow_nan** is True. If allow_nan is False, then it will be a ValueError to serialize out of range float values (nan, inf, -inf) in strict compliance with the JSON specification, instead of using the JavaScript equivalents (NaN, Infinity, -Infinity).

- If **indent** is a non-negative integer or string, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0, negative, or "" will only insert newlines. None (the default) selects the most compact representation. Using a positive integer indent indents that many spaces per level. If indent is a string (such as "\t"), that string is used to indent each level.
- Use (',', ': ') as **default** if indent is not None.
- The default value of **sort_keys** is False. If sort_keys is True, then the output of dictionaries will be sorted by key.

# Decode JSON strings into Python objects

```
json.load(fp,
          cls=None,
                  object_hook=None,
                  parse_float=None,
                  parse_int=None,
                  parse_constant=None,
                  object_pairs_hook=None, **kw)
```

The above method deserialize fp (a .read()-supporting file-like object containing a JSON document) to a Python object using the following conversion table.

| JSON | Python |
|---|---|
| object | dict |
| array | list |
| string | str |
| number (int) | int |
| number (real) | float |
| true | True |
| false | False |
| null | None |

# Options :

- The default value of **skipkeys** is False. If skipkeys is True, then dict keys that are not of a basic type (str, int, float, bool, None) will be skipped instead of raising a TypeError.
- The json module always produces str objects, not bytes objects. Therefore, fp.write() must support str input.
- If ensure_ascii is True (the default), the output is guaranteed to have all incoming non-ASCII characters escaped. If ensure_ascii is False, these characters will be output as-is.
- The default value of check_circular is True. If check_circular is False, then the circular reference check for container types will be skipped and a circular reference will result in an OverflowError.
- The default value of allow_nan is True. If allow_nan is False (default: True), then it will be a ValueError to serialize out of range float values (nan, inf, -inf) in strict compliance with the JSON specification, instead of using the JavaScript equivalents (NaN, Infinity, -Infinity).

- If indent is a non-negative integer or string, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0, negative, or "" will only insert newlines. None (the default) selects the most compact representation. Using a positive integer indent indents that many spaces per level. If indent is a string (such as "\t"), that string is used to indent each level.
- Use (',', ': ') as default if indent is not None.
- default(obj) is a function that should return a serializable version of obj or raise TypeError. The default simply raises TypeError.
- The default value of sort_keys is False. If sort_keys is True, then the output of dictionaries will be sorted by key.

# Python and XML

# XML

- The Extensible Markup Language (XML) is a markup language much like **HTML** or **SGML (standard generalized markup language**).
- This is recommended by the World Wide Web Consortium and available as an open standard.
- XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

# XML Parser Architectures and APIs

- **A parser is a compiler or interpreter component that breaks data into smaller elements for easy translation into another language.**
- The Python standard library provides a minimal but useful set of interfaces to work with XML.
- **The two most basic and broadly used APIs** to XML data are the **SAX** and **DOM** interfaces.
- **Simple API for XML (SAX) :**
- Here, you **register callbacks for events of interest** and **then let the parser proceed through the document.**
- This is **useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.**
- **Document Object Model (DOM) API :**
- This is a World Wide Web Consortium recommendation wherein the **entire file is read into memory** and **stored in a hierarchical (tree-based) form to represent all the features of an XML document.**

- SAX obviously cannot process information as fast as DOM can when working with large files.
- On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.
- SAX is read-only, while DOM allows changes to the XML file.
- Since these two different APIs literally complement each other, there is no reason why you can not use them both for large projects.

# Parsing XML with SAX APIs

- SAX is a standard interface for event-driven XML parsing.
- Parsing XML with SAX generally requires you to create your own ContentHandler by subclassing **xml.sax.ContentHandler.**
- Your *ContentHandler handles the particular tags and attributes of your flavor(s) of XML.*
- *A **ContentHandler object provides methods to handle various parsing events**.*
- *Its owning parser calls ContentHandler methods as it parses the XML file.*
- **The methods *startDocument* and *endDocument* are called at the start and the end of the XML file.**
- ***The method characters(text) is passed character data of the XML file via the parameter text.***

- The ContentHandler is called at the start and end of each element.
- If the parser is not in namespace mode, the methods *startElement(tag, attributes)* and *endElement(tag)* are called;
- *otherwise, the corresponding methods startElementNS and endElementNS are called.*
- *Here, tag is the element tag, and attributes is an Attributes object.*

# The *make_parser* Method

- Following method creates a new parser object and returns it.
- The parser object created will be of the first parser type the system finds.
- **xml.sax.make_parser( [parser_list] )**
- **parser_list:** The optional argument consisting of a list of parsers to use which must all implement the make_parser method.
-

# parseMethod

- Following method creates a SAX parser and uses it to parse a document.

**xml.sax.parse( xmlfile, contenthandler[, errorhandler])**

- **Here is the detail of the parameters:**
- **xmlfile:** This is the name of the XML file to read from.
- **contenthandler:** This must be a ContentHandler object.
- **errorhandler:** If specified, errorhandler must be a SAX ErrorHandler object.

# parseStringMethod

- There is one more method to create a SAX parser and to parse the specified **XML string.**

xml.sax.parseString (xmlstring, contenthandler[, errorhandler])

- **Here is the detail of the parameters:**
- **xmlstring:** This is the name of the XML string to read from.
- **contenthandler:** This must be a ContentHandler object.
- **errorhandler:** If specified, errorhandler must be a SAX ErrorHandler object.

# Parsing XML with DOM APIs

- The **Document Object Model (DOM)** is a cross-language API from the World Wide Web Consortium (W3C) **for accessing and modifying XML documents.**
- The DOM is extremely **useful for random-access applications.**
- SAX only allows you a view of one bit of the document at a time. **If you are looking at one SAX element, you have no access to another.**
- Here is the **easiest way to quickly load an XML document and to create a minidom object using the xml.dom module.**
- The **minidom object provides a simple parser method that quickly creates a DOM tree from the XML file.**
- The sample phrase calls the **parse( file [,parser] ) function of the minidom object to parse the XML file designated by file into a DOM tree object.**